Architecting a Distributed Dynamic Image Server for the Web

Alain Chesnais, Tim Beck, Rudy Ziegler
TrueSpectra, Inc
http://www.truespectra.com/

Introduction

One of the most labor-intensive tasks in the development of web sites is the creation of derivative images for display of a visual element at multiple sizes or in multiple styles. For instance, a typical online catalog will have at least three different visual representations for each product sold: one thumbnail sized image for visual browsing, one mid sized image for viewing a product description and one large sized image for viewing product detail. The approach outlined in this presentation is to enable the web server to actively generate any derivative image from a high-resolution base image. Derivative images are then cached on the server to speed delivery of subsequent requests for the same derived content.

Dynamic Image Serving

We call the process of enabling the server to generate derivative images on the fly "Dynamic Image Serving". In the implementation described in this presentation, commands are passed to the server via standard HTTP query strings.

Here are two examples with their associated URLs.

Thumbnail:
http://iris.truespectra.com/overview/phone.jpg?**wid=75**&cvt=jpeg

Medium:
http://iris.truespectra.com/overview/phone.jpg?**wid=175**&
cvt=jpeg

The WID parameter that is passed to the base image in the examples above controls the width of the resulting image in pixels

The overall effect of doing this is to enable a web server to become aware of image assets and automatically perform image manipulation without requiring the use of CGI scripts or back end coding.

Dynamic Image Serving Architecture

We propose to describe an architecture for implementing Dynamic Image Serving based upon three components:

- A cache component responsible for storing results that have already been calculated
- A render component responsible for actually doing the image manipulations that are requested
- An image store component responsible for storing the high-resolution base images and delivering the relevant portions of these images to the render component.

The challenge that arises when deploying such a solution is determining how to be able to scale as demand on the server rises. Scaling can occur in three areas: number of total hits, number of render requests and number of base images to store.

The architecture that we present allows you to implement each component described above on dedicated machines. We will describe how to monitor and scale each set of machines based on the observed load. Appropriate metrics are developed and described to show how one can then scale adequately in either of three ways (total hits, render requests, number of images) and maintain an optimally performing system. Examples are taken from our experience in providing such a networked solution in a geographically distributed Dynamic Image Serving network for the past year.